These tutorials are a simplified introduction, and are not sufficient on their own to achieve system safety. You are responsible for the safety of your system.

"Adventure is just bad planning."
– Roald Amundsen
Anti-Patterns for Safety Plans:
- It’s just a pile of unrelated documents
- It doesn’t address software integrity
- You don’t link to a relevant safety standard
- It doesn’t link to a security plan

Safety Plan:
- Safety Standard: pick a suitable standard
- Hazards & Risks: hazard log, criticality analysis
- Goals: safety strategy, safety requirements
- Mitigation & Analysis: HAZOP, FMEA, FTA, ETA, reliability, ...
- Safety Case: safety argument
Usually "functional safety" (safety functions)
- IEC 61508 is a generic starting point
- Many domains have specific standards
  - ISO 26262, EN-50126/8/9, MIL-STD-882, IEC 60730, DO-178, ...

Key elements of a safety standard:
- Method for determining risk
  - Usually Safety Integrity Level (SIL)
- SIL determines engineering rigor
  - Analysis techniques
  - Mitigation techniques
- Life-cycle approach to safety
Safety Goals & Safety Requirements

Safety Goal: top level definition of “safe”
- Example: vehicle speed control
  - Hazard: unintended vehicle acceleration
  - Goal: engine power proportional to accel. pedal position
- Safety strategy: how you plan to achieve goal
  - Example: correct computation AND engine shutdown if unintended acceleration

Safety Requirements:
- Goals at system level; requirements provide supporting detail
- Supporting requirements generally allocated to subsystems
  - Might include functionality and fail-safe mitigation requirements
- Examples:
  - Engine torque shall match accelerator position torque curve
  - Pedal/torque mismatch shall result in engine shutdown
FMEA: Failure Mode Effects Analysis

Idea: Start with component failure; analyze results; identify hazards

<table>
<thead>
<tr>
<th>Component</th>
<th>Potential Failure Mode</th>
<th>Failure Effects</th>
<th>Recommended Action</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistor R2</td>
<td>Open</td>
<td>Triggers Shutdown</td>
<td>Use Industrial spec. component</td>
<td>Done</td>
</tr>
<tr>
<td>Short</td>
<td></td>
<td>Over-current/potential Fire</td>
<td>Circuit Redesign</td>
<td>Open</td>
</tr>
<tr>
<td>Capacitor C7</td>
<td>Explodes</td>
<td>Potential Fire</td>
<td>Select different component</td>
<td>Open</td>
</tr>
</tbody>
</table>

Significant limitations for generating hazards

- “Complex component” failures are not well behaved
  - Software fails however it wants to fail
  - Integrated circuits are usually highly coupled internally
- Poor at representing correlated and accumulated faults
  - E.g., exploding capacitor damaging several nearby components
HAZard and Operability Analysis (HAZOP)

Hazard structured brainstorming

- For each system requirement:
  - Modify with a guide word
  - Does the result suggest a hazard?
- Effective starting point, but not guaranteed to find all hazards

Examples

- When pressure exceeds 6000 psig, relief valve shall **NOT** actuate.
- System shall come to a complete stop within **AFTER** 5 seconds when emergency stop is activated.
  - Alternately: System shall come to a complete stop within 5 seconds **LATE** when emergency stop is activated.
Hazard: a potential source of injury or damage
- A potential cause of a mishap or loss event (people, property, financial)

Hazard log
- Captures hazards for a system
- HAZOP generates some hazards
- Others are legacy & experience

Risk evaluation
- Risk = Probability * Consequence
  - Typically determined via a risk table
- Risk must be reduced to acceptable levels
  - Risk determines required SIL (e.g. “Very High” → SIL 4)
Safety Analysis & Mitigation

- **Failure Mode Effects Analysis (FMEA)**
  - Work forward from fault to mishap

- **Fault Tree Analysis (FTA)**
  - Work backward from hazard to causes
  - **Strategy**: HAZOP identifies fault tree roots

- **Avoid single points of failure**
  - If component breaks, is system unsafe?
  - Computational elements fail in worst way

- **Life-critical systems require redundancy**
  - Also avoid correlated faults
  - High-SIL software techniques to avoid SW defects
This system is safe because: structured argument + evidence

Incorporates safety plan topics:
- Methodical identification of hazards
- Each hazard evaluated for risk
- Mitigation rigor determined by risk (e.g., SIL)
- Analysis rigor determined by risk (e.g., SIL)
- Safety requirements appropriately cover all hazards
  - Including both accidental faults & malicious faults

Example techniques
- Goal Structuring Notation (GSN)  
- Systems-Theoretic Process Analysis (STPA / Leveson)

[GSN Standard]
Best Practices For Safety Plans

- A written Safety Plan including:
  - Hazards + risks
  - Safety goals + requirements
  - Safety analysis + Mitigation
  - Following a safety standard
  - Resulting in a written safety case
  - Independent audit of safety case

- Pitfalls:
  - Software safety usually stems from rigorous SIL engineering
  - FMEA can miss correlated & multipoint faults – must use FTA
  - Need to include safety caused by security attacks

https://www.flickr.com/photos/jurvetson/1118807